



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

09/537,998

03/29/2000

David D'Souza

MSFT115144

6844

26389

7590

02/25/2004

EXAMINER

LAO, SUE X

CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC  
1420 FIFTH AVENUE  
SUITE 2800  
SEATTLE, WA 98101-2347

ART UNIT

PAPER NUMBER

2126

DATE MAILED: 02/25/2004

3

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

09/537,998

Applicant(s)

D'SOUZA, DAVID

Examiner

S. Lao

Art Unit

2126

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☐ Responsive to communication(s) filed on \_\_\_\_.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-15 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-15 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. §§ 119 and 120**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).  
a) ☐ All b) ☐ Some \* c) ☐ None of:  
1. ☐ Certified copies of the priority documents have been received.  
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_.  
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).  
\* See the attached detailed Office action for a list of the certified copies not received.
- 13) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.  
a) ☐ The translation of the foreign language provisional application has been received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 2.
- 4) ☐ Interview Summary (PTO-413) Paper No(s). \_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_.

### DETAILED ACTION

1. Claims 1-15 are presented for examination.
2. The non-statutory double patenting rejection, whether of the obviousness-type or non-obviousness-type, is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent. *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); and *In re Goodman*, 29 USPQ2d 2010 (Fed. Cir. 1993).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(b) and © may be used to overcome an actual or provisional rejection based on a non-statutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.78(d).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1-15 are rejected under the judicially created doctrine of obviousness - type double patenting as being unpatentable over claims 1-14 of U.S. Patent No. 6,052,707 to D'Souza. Although the conflicting claims are not identical, they are not patentably distinct from each other.

As to claim 1, U.S. Patent No. 6,052,707 teaches in a data processing system having a least one processor for running tasks (tasks) and resources (common module of code) available to tasks, a method comprising the steps of:

logically partitioning tasks into groups of tasks that utilize the same resources (claim 1, lines 6-8);

preemptively scheduling the groups of tasks to be run such that each group of tasks is given a time slot in a cycle in which its tasks may run on the processor (claim 1, lines 9-12); and

for each group, non-preemptively scheduling tasks to be run within each group during the time slot allocated to the group (claim 1, lines 13-15).

As to claim 2, U.S. Patent No. 6,052,707 teaches the data processing system includes at least one storage device and the method further comprises the step of storing a group list for each associated group in the storage device, wherein each group list includes identifying information for tasks included in the associated group (claim 2, lines 1-6).

As to claim 3, U.S. Patent No. 6,052,707 teaches storing status information for each group indicating whether the group has a task that is running and holding identifying information about any task that is running (claim 3, lines 1-4).

As to claim 4, U.S. Patent No. 6,052,707 teaches the step of logically partitioning tasks into groups of tasks that utilize the same resources further comprises the steps of: initially placing each task in its own group; and subsequently combining groups of tasks into a merged group (moving into another group) wherein each task in the merged group allocates a common resource when run (claim 4, lines 4-7).

As to claim 5, note discussion of claim 1 and U.S. Patent No. 6,052,707 further teaches a data processing system having at least one storage device for storing modules of code, at least one processor for running tasks wherein running each task involves allocating at least one resource and resources that may be allocated to tasks (claim 5, lines 1-4), a method comprising the steps of:

providing a task dependency list for each task, said task dependency list listing resources that are candidates to be allocated when the task is run on the processor (claim 5, lines 6-8);

examining the task dependency lists to logically partition the tasks into groups of interdependent tasks (claim 5, lines 9-17; claim 8, lines 14-15).

As to claims 6, 7, note discussion of claims 2 and 3, respectively.

As to claim 8, U.S. Patent No. 6,052,707 teaches in a data processing system having at least one storage device for storing modules of code and at least one processor for running tasks, and resources that may be allocated by tasks, wherein running each task involves allocating at least one resource, a method comprising the steps of:

- providing a module and resource dependency list for each associated module of code, wherein each module and resource dependency list lists interdependent modules of code of the associated module of code and resources allocated by the associated module of code (claim 8, lines 6-9);

- generating a task dependency list for each task by taking a logical union of modules and resources listed in the module and resource dependency lists of modules and resources that are candidates to be called when the task is run on the processor (claim 8, lines 10-13);

- examining the task dependency lists to logically partition the tasks into groups of interdependent tasks (claim 8, lines 14-15); and

- preemptively scheduling the groups of tasks to be run such that each group of tasks is given a time slot in a cycle in which its tasks may run on the processor (claim 8, lines 16-18);

- and for each group of tasks, non-preemptively scheduling tasks to be run within each group during the time slot allocated to the group (claim 8, lines 19-21).

As to claims 9, 10, note discussion of claims 2 and 3, respectively.

As to claim 11, U.S. Patent No. 6,052,707 teaches a data processing system, comprising:

- a partitioning mechanism for partitioning tasks into groups of interdependent tasks that utilize the same resources (claim 1, lines 6-8. Using a partitioning mechanism to perform this step would have been obvious);

- an execution mechanism for executing the tasks (processor, claim 1, lines 1-2);

- a preemptive scheduler for preemptively scheduling the groups of tasks such that each group is given a time slot in which to execute one of its tasks (claim 1, lines 9-12. Using a scheduler means to perform this step would have been obvious); and

a non-preemptive scheduler for non-preemptively scheduling tasks within each group (claim 1, lines 13-15. Using a scheduler means to perform this step would have been obvious).

As to claim 12, U.S. Patent No. 6,052,707 teaches in a data processing system having at least one processor for running tasks and resources that may be allocated by said tasks, wherein said processor runs an operating system, a computer-readable storage medium holding the operating system, said operating system performing the steps of:

logically partitioning tasks into groups of interdependent tasks to be run non-preemptively, said tasks allocating the same resources, and tasks that do not allocate the same resources being placed as separate groups (claim 1, lines 6-8, 13-15; claim 5, lines 12-17; claim 11, lines 7-8);

preemptively scheduling the groups of tasks to be run such that each group of tasks is given a time slot in a cycle in which its tasks may run on the processor (claim 1, lines 9-12); and

for each group, non-preemptively scheduling tasks to be run within each group during the time slot allocated to the group (claim 1, lines 13-15).

As to claim 13, note discussion of claim 4.

As to claim 14, U.S. Patent No. 6,052,707 teaches in a data processing system having at least one storage device for storing modules of code, system resources that may be allocated by tasks, and at least one processor for running tasks wherein running each task involves allocating at least one resource, a computer-readable storage medium holding an operating system for performing the steps of:

providing a task dependency list for each task, said task dependency list listing resources that are allocated when the task is run on the processor (claim 5, lines 6-8);

examining the task dependency lists to logically partition the tasks into groups of interdependent tasks such that, for each task, modules on the task dependency list for the task are included in a single group (claim 8, lines 14-15; claim 5, lines 9-17);

preemptively scheduling the groups of tasks to be run such that each group of tasks is given a time slot in a cycle in which its tasks may run on the processor (claim 1, lines 9-12); and

for each group of tasks, non-preemptively scheduling tasks to be run within each group during the time slot allocated to the group (claim 1, lines 13-15).

As to claim 15, U.S. Patent No. 6,052,707 teaches in a data processing system having at least one storage device for storing modules of code, system resources that may be allocated by tasks, and at least one processor for running tasks wherein running each task involves allocating at least one resource, a computer-readable storage medium holding an operating system for performing the steps of:

providing a module and resource dependency list for each associated module of code, wherein each module and resource dependency list lists interdependent resources (modules of code) of the associated module of code (claim 14, lines 7-10);

generating a task dependency list for each task by taking a logical union of resources listed in the module and resource dependency lists of resources that are allocated when the task is run on the processor (claim 14, lines 11-14);

examining the task dependency lists to logically partition the tasks into groups of interdependent tasks (claim 14, lines 15-18);

preemptively scheduling the groups of tasks to be run such that each group of tasks is given a time slot in a cycle in which its tasks may run on the processor (claim 14, lines 19-21); and

for each group of tasks, non-preemptively scheduling tasks to be run within each group during the time slot allocated to the group (claim 14, lines 22-24).

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claims 12, 13 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 12 recites "logically partitioning tasks into groups of interdependent tasks to be run non-preemptively" in lines 5-6. It is not clear whether it is the groups or the tasks that are to be run non-preemptively.

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

7. Claims 1-4, 11-13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Prosise, Pietrek and Stinson.

It is noted that Prosise and Pietrek discuss features of Windows 3.1 and Stinson is cited to show that Windows 3.1 was available back in 1992 and therefore the teaching of Prosise and Pietrek was available back in 1992.

As to claim 1, Prosise teaches tasks (Windows applications, DOS applications), logically partitioning tasks into groups of interdependent tasks (group of Windows applications to be run in System VM, and group of DOS applications to be run in DOS VMs), preemptively scheduling the groups (preemptive scheduler, divides CPU time among System VM and DOS VMs preemptively), each group is given a time slot (time slicing), non-preemptively scheduling within a group (non-preemptive scheduler associated with Windows applications within System VM), see fig. 1; pg. 261, left col., 2nd para.; pg. 263, right col., section "Windows Does It Too" to pg. 264, left col..

It is noted that Windows OS and DOS each provides its own system routines, ie, resources, to be utilized/called by application programs running thereon. Windows applications grouped under System VM call/utilize Windows system routines /



resources, and DOS applications grouped under DOS VMs call/utilize DOS system routines / resources.

Prosise does not explicitly teach that the non-preemptively scheduling is performed within *each* of all the groups, which, however, would have been an obvious choice so as to provide simplicity and consistency in scheduling policies.

As to claim 2, Pietrek teaches (pg. 225, 3rd and last para.s) storing a group list for each associated group (task database TDB, maintained in a linked list), identifying information for tasks (selector of next TDB).

As to claim 3, Pietrek teaches (TDB, table 3-2) storing status information (flags, error flags, etc.), whether the group has a task that is running (DLL is loading, DLL is unloading), holding identifying information about any task that is running (module handle for task). It is noted that an application program conventionally comprises some code and therefore running a task/application would involve running a module of code.

As to claim 4, Prosise as modified teaches initially placing each task in its own group (a Windows application), subsequently combining groups of tasks into a merged group (Windows applications under System VM, Prosise, fig. 1), each task in the merged group calls a common resource (share code segments and resources; share resources by every user of a module, see Pietrek, pg. 215, 3rd para.; pg. 217, 4th para.). It is noted that moving the initially placed task into another group would have been an obvious way to achieve combining.

As to claim 11, note discussion of claim 1 for partitioning, preemptively scheduling and non-preemptively scheduling. Using a partitioning mechanism to perform partitioning, a preemptive scheduler to perform preemptive scheduling and a non-preemptive scheduler to perform non-preemptive scheduling would have been obvious. Prosise and Pietrek further teaches execution mechanism for executing the tasks (processor).

As to claim 12, note discussion of claim 1 for preemptively scheduling and non-preemptively scheduling. Prosise and Pietrek further teaches operating system (Windows 3.x, Windows 3.1), logically partitioning tasks into groups of interdependent tasks to be run non-preemptively (group of Windows applications to be run in System

VM, and group of DOS applications to be run in DOS VMs), and tasks allocating the same resources, and tasks that do not allocate the same resources being placed as separate groups in that Windows OS and DOS each provides its own system routines, ie, resources, to be allocated/called by application programs running thereon. Windows applications grouped under System VM allocate/utilize Windows system routines / resources, and DOS applications grouped under DOS VMs allocate/utilize DOS system routines / resources.

As to claim 13, note discussion of claim 4.

8. Claims 5-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Prorise, Pietrek and Stinson as applied to claim 1 and further in view of Tulpule et al.

As to claim 5, it is covered by claim 1 except for (1) modules of code, (2) providing a task dependency list for each task, the task dependency list listing resources that are candidates to be allocated when the task is run, (3) that the logically partitioning includes examining the task dependency lists to group the tasks into groups of interdependent tasks.

As to (1)-(2), Pietrek teaches modules of code (module, code, pg. 213, last para.), providing a task dependency list for each task (task database TDB), the task dependency list listing resources that are candidates to be allocated when the task is run (OpenAppEnv() creates a segment that contains the module handles of all the DLLs of the loading executable module; pg. 252, 1st para.; module handle for task; Table 3-2).

As to (3), Prorise teaches interdependent tasks (windows applications) are grouped together (grouped to be run under System VM), wherein the modules run by the tasks (windows resources, windows DLLs) are included in a single group (under Windows). Pietrek teaches describing modules related to a task with a task dependency list, and thus the combined teaching of Prorise and Pietrek would have provided the Windows and DOS applications with task dependency lists. Tulpule teaches partitioning tasks based on interdependencies, including examining task dependency lists (Tasks are scheduled in a plurality of task execution queues. A task of highest priority, which

Art Unit: 2126


has met all its prerequisites, is searched and invoked for execution. See col. 4, lines 19-22; col. 6, lines 6-13). One of ordinary skill in the art would have been motivated to combine the teachings of Prosise and Pietrek with Tulpule because this would have provided a flexible configuration to dynamically respond to changes of task's execution times (Tulpule, col. 2, lines 29-34).

As to claims 6 and 7, note discussions of claims 2 and 3, respectively.

As to claim 14, note discussion of claim 5. Further, Prosise and Pietrek further teaches operating system (Windows 3.x, Windows 3.1), system resources (Windows system routines, DOS system routines). Prosise and Pietrek teaches for each task, modules on the task dependency list for the task are included in a single group in that Windows OS and DOS each provides its own system routines, ie, resources, to be allocated/called by application programs running thereon. Windows applications grouped under System VM allocate/utilize Windows system routines / modules, and DOS applications grouped under DOS VMs allocate/utilize DOS system routines / modules.

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Sue Lao whose telephone number is (703) 305-9657. A voice mail service is also available at this number. The examiner's supervisor, SPE Meng-Ai An, can be reached on (703) 305 9678. The examiner can normally be reached on Monday - Friday, from 9AM to 5PM. The fax phone number for the organization where this application or proceeding is assigned is (703) 872 9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-9600.

Sue Lao   
February 19, 2004